

主讲教师：传智.神龙教主

传智播客 PHP学院 <http://php.itcast.cn> 发布

## 回顾

1. MVC模式
   1. M：model(模型)，用来操作数据
   2. V：view(视图)，用来显示数据
   3. C：Controller(控制器)，用来做业务逻辑

每次请求都是请求控制器，控制器根据需要调用数据（模型）和视图

1. 一个表就是一个模型。命名：表名+Model
2. 一个模块就是一个控制器。命名：模块名+Controller
3. 一个文件中存放一个类，文件名以.class.php结尾
4. 所有的类文件的文件名必须和类名一致

## netbeans的介绍

netbeans工具是一个非常优秀操作PHP的工具之一（还有一个phpstorm），



### 安装netbeans

要安装netbeans必须先安装jdk(java开发包)，

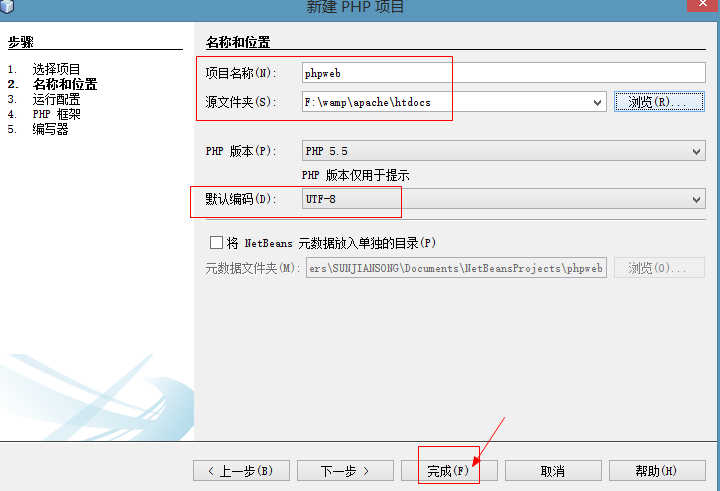
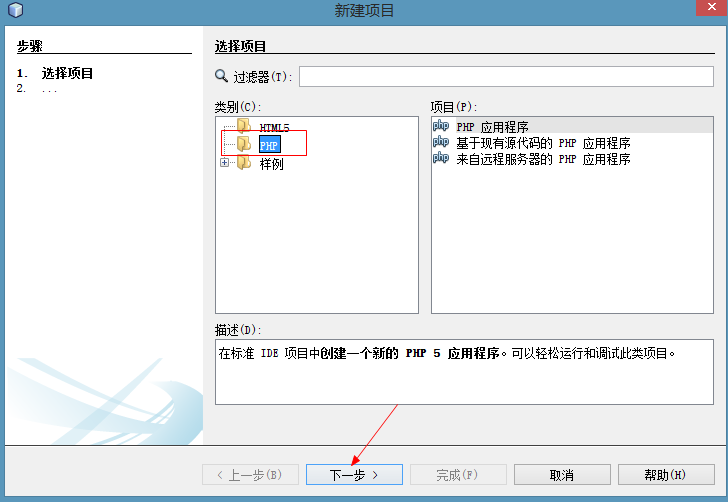
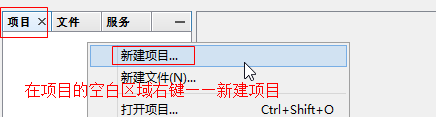


安装好jdk后，直接安装netbeans



下一步、下一步安装即可

### 使用netbeans



项目创建完成后，会在虚拟目录下多一个“nbproject”文件夹



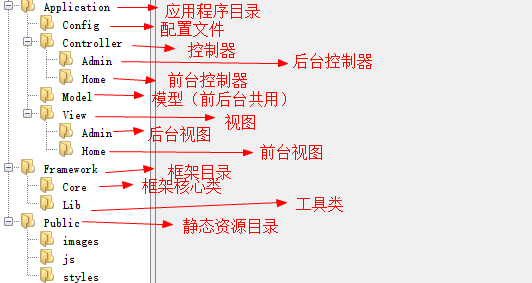
脚下留心：nbproject文件夹不是项目的组成部分，此文件夹就是用来管理虚拟目录，在站点上传的时候把此文件夹删除。

## 框架搭建

一个项目的代码可以分成两个部分：一部分代码用来保证程序的运行（框架代码），一部分代码用来执行业务逻辑（项目代码）。

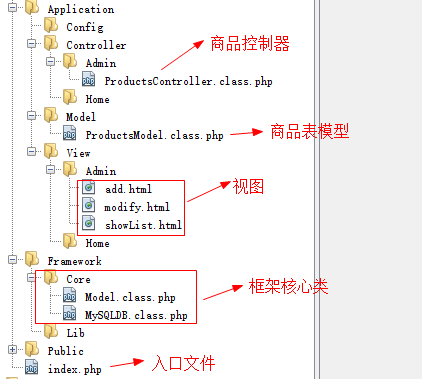
一个框架上可以支持多个项目

### 框架目录结构



将上一讲的文件分类摆放到对应文件夹下。

分类摆放上一讲的文件。如下图

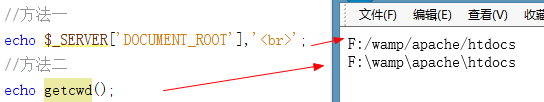


分类摆放后，程序无法运行，原因是无法加载需要的类。下面的重点就是如何实现自动加载类

脚下留心：因为框架每次请求都是入口文件（index.php）,那么在框架的（.）表示当前入口文件所在的目录。

### 知识点

1. 获取入口文件所在的目录



多学一招：物理地址的分隔用”\”,网址的分隔用“/”。在Linux中严格区分；在windows中，表示物理地址这两种分隔符都可以

1. DIRECTORY\_SEPARATOR常量通过判断当前操作系统来确定目录分隔符。



### 配置文件

在Application/Config文件夹下新建一个config.php页面，此页面中存放配置信息

<?php

//config.php配置文件前面要注意有return

return array(

//数据库的配置信息

'database'=>array(

'host' => '127.0.0.1',

'port' => '3306',

'user' => 'root',

'pwd' => 'aa',

'charset'=> 'utf8',

'dbname'=> 'php4'

),

//应用程序的配置信息

'app' =>array(

'default\_platform' => 'Admin',

'default\_controller'=> 'Products',

'default\_action' => 'list',

)

);

### 定义常量

/\*

\* 定义路径常量

\*/

private static function initConst(){

define('DS', DIRECTORY\_SEPARATOR); //定义目录分隔符

define('ROOT\_PATH', getcwd().DS); //根目录

define('APP\_PATH', ROOT\_PATH.'Application'.DS); //Application目录

define('FRAMEWROK\_PATH', ROOT\_PATH.'Framework'.DS); //Framework目录

define('PUBLIC\_PATH', ROOT\_PATH.'Public'.DS); //Public目录

define('CONFIG\_PATH', APP\_PATH.'Config'.DS); //Config目录

define('CONTROLLER\_PATH', APP\_PATH.'Controller'.DS); //Controller目录

define('MODEL\_PATH', APP\_PATH.'Model'.DS); //Model目录

define('VIEW\_PATH', APP\_PATH.'View'.DS); //View目录

define('CORE\_PATH', FRAMEWROK\_PATH.'Core'.DS); //Core目录

define('LIB\_PATH', FRAMEWROK\_PATH.'Lib'.DS); //Lib目录

}

### 导入配置文件

/\*

\* 导入配置文件

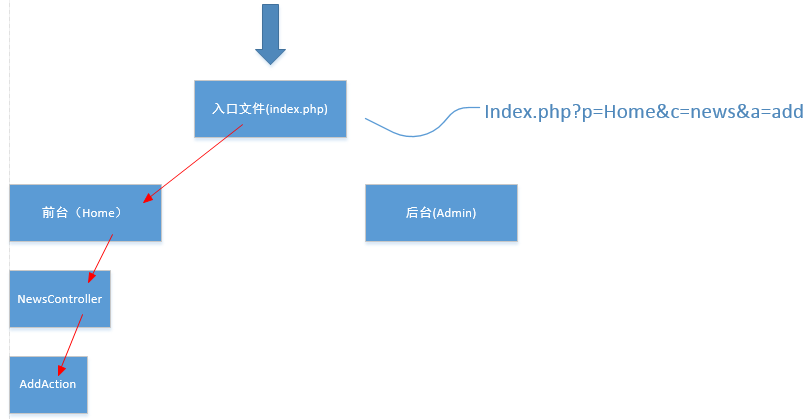
\*/

private static function initConfig(){

$GLOBALS['config']=require CONFIG\_PATH.'config.php';

}

### 确定路由



/\*

\* 确定路由

\*/

private static function initRoutes(){

$p=isset($\_REQUEST['p'])?$\_REQUEST['p']:$GLOBALS['config']['app']['default\_platform']; //当前平台名

$c=isset($\_REQUEST['c'])?$\_REQUEST['c']:$GLOBALS['config']['app']['default\_controller'];//当前控制器名

$a=isset($\_REQUEST['a'])?$\_REQUEST['a']:$GLOBALS['config']['app']['default\_action'];//当前方法名

define('PLATFORM\_NAME', $p);

define('CONTROLLER\_NAME', $c);

define('ACTION\_NAME', $a);

define('\_\_URL\_\_', CONTROLLER\_PATH.PLATFORM\_NAME.DS); //当前控制器的目录

define('\_\_VIEW\_\_', VIEW\_PATH.PLATFORM\_NAME.DS); //当前平台的目录

}

### 自定义加载类

/\*

\* 自定义自动加载类

\*/

private static function autoLoad($class\_name){

$class\_map=array(

'MySQLDB' => CORE\_PATH.'MySQLDB.class.php',

'Model' => CORE\_PATH.'Model.class.php',

'Controller'=> CORE\_PATH.'Controller.class.php'

);

if(isset($class\_map[$class\_name])){

require $class\_map[$class\_name];

}

elseif(substr($class\_name, -5)=='Model'){

require MODEL\_PATH.$class\_name.'.class.php';

}

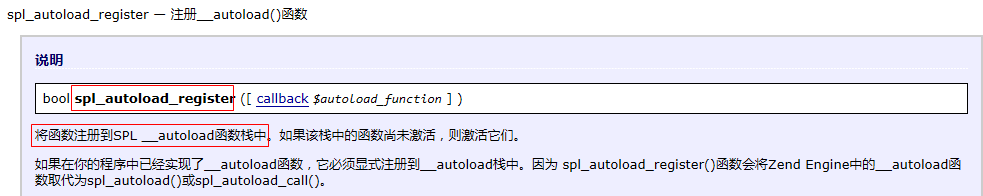
elseif(substr($class\_name, -10)=='Controller'){

require \_\_URL\_\_.$class\_name.'.class.php';

}

}

### 注册自动加载类



/\*

\* 注册自定义加载类函数

\*/

private static function initRegisterAutoLoad(){

spl\_autoload\_register('self::autoLoad');

}

### 请求分发

/\*

\* 请求分发

\*/

private static function initDispatch(){

$controller\_name=CONTROLLER\_NAME.'Controller'; //拼接控制器名

$action\_name=ACTION\_NAME.'Action'; //拼接方法名

$controller=new $controller\_name(); //实例化控制器

$controller->$action\_name(); //调用方法

}

### run()方法

运行框架

public static function run(){

self::initConst();

self::initConfig();

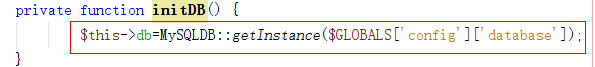
self::initRoutes();

self::initRegisterAutoLoad();

self::initDispatch();

}

脚下留心：在基础模型中，更改获取MySQLDB类的实例



### 运行框架

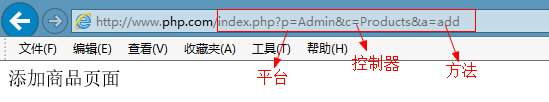
在入口文件(index.php)页面，引入框架文件并调用run()方法

<?php

require './Framework/Core/Framework.class.php';

Framework::run();

### 测试页面



## 完善删除功能

### 点击“删除”按钮



<input type="button" value="删除" onclick="if(confirm('确定要删除吗'))location.href='index.php?p=Admin&c=Products&a=del&id=<?php echo $rows['proID'];?>'">

### 跳转方法

方法一：js的定时器

<script type="text/javascript">

window.onload=function(){

setTimeout(function(){

location.href='http://www.php.com/index.php';

},3000);

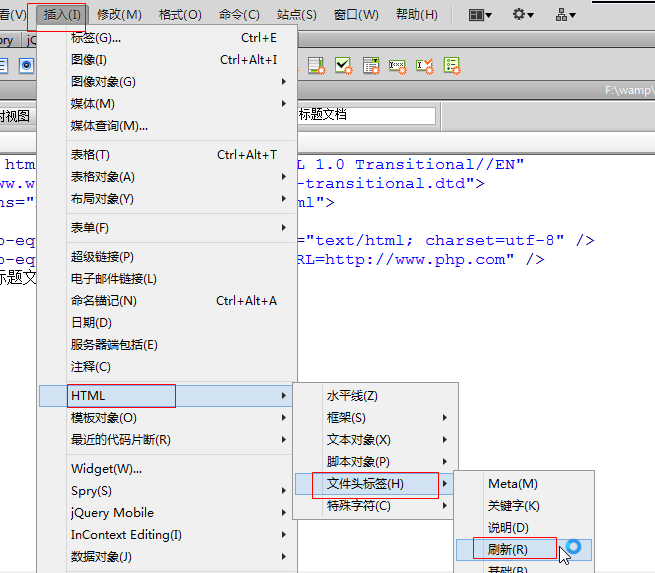
}

</script>

方法二：<meta>

<meta http-equiv="refresh" content="3;http://www.php.com" />

可以通过Dreamweaver软件自动生成上面的代码



在基础控制器中写页面跳转的方法

<?php

/\*

\* 基础控制器类

\*/

class Controller{

/\*

\* 成功跳转的方法

\* @param $url string 跳转的地址

\* @param $msg string 显示信息，如果$msg为空，就直接跳转

\* @param $time int 页面停留时间

\*/

public function success($url,$msg='',$time=3){

$this->jump($url,$msg,$time,true);

}

/\*

\* 失败的跳转

\*/

public function error($url,$msg='',$time=3){

$this->jump($url,$msg,$time,false);

}

/\*

\* 跳转的方法

\*/

private function jump($url,$msg='',$time=3,$flag=true){

if($msg==''){

header("location:{$url}");

}else{

if($flag)

$path='<img src="/Public/images/success.jpg" />';

else

$path='<img src="/Public/images/error.jpg" />';

echo <<<jump

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta http-equiv="refresh" content="{$time};URL={$url}" />

<title>无标题文档</title>

<style type="text/css">

body{

text-align:center;

font-size:20px;

background-color:#F90;

color:#F00;

padding-top:30px;

font-family:'微软雅黑'

}

</style>

</head>

<body>

{$path}

{$msg}

</body>

</html>

jump;

}

}

}

### 调用跳转的方法

在控制器中，如果成功就调用success()，失败就调用error()方法

